Name: Mohit Kerkar Div: D15C Roll No: 23

<u>Adv DevOps Exp 12</u>

<u>Aim</u>: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3

Theory:

AWS Lambda and S3 Integration:

AWS Lambda allows you to execute code in response to various events, including those triggered by Amazon S3. When an object is added to an S3 bucket, it can trigger a Lambda function to execute, allowing for event-driven processing without managing servers.

Workflow:

1. Create an S3 Bucket:

○ First, create an S3 bucket that will store the objects. This bucket will act as the trigger source for the Lambda function.

2. Create the Lambda Function:

○ Set up a new Lambda function using AWS Lambda’s console. You can choose a runtime environment like Python, Node.js, or Java.

○ Write code that logs a message like “An Image has been added” when triggered. 3. Set Up Permissions:

○ Ensure that the Lambda function has the necessary permissions to access S3. You can do this by attaching an IAM role with policies that allow reading from the bucket and writing logs to CloudWatch.

4. Configure S3 Trigger:

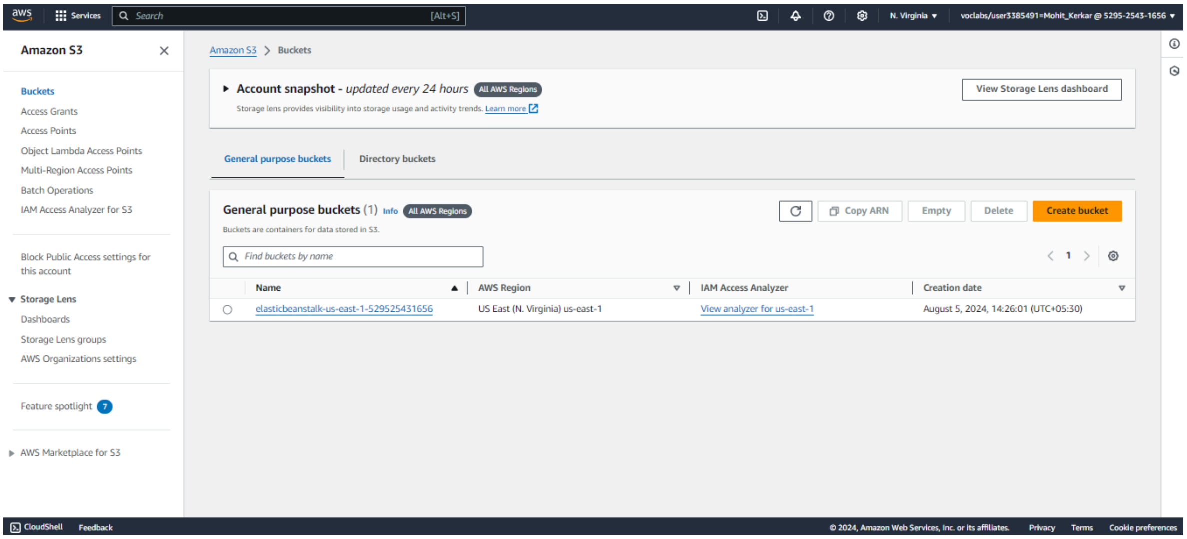
○ Link the S3 bucket to the Lambda function by setting up a trigger. Specify that the function should be triggered when an object is created in the bucket (e.g., when an image is uploaded).

5. Test the Setup:

○ Upload an object (e.g., an image) to the S3 bucket to test the trigger. The Lambda function should execute and log the message “An Image has been added” in AWS CloudWatch Logs.

Step 1: S3 bucket creation

Navigate to the S3 inside services section on AWS and create a new bucket



Account snapshot - *updated every 24 hours* All AWS Regions

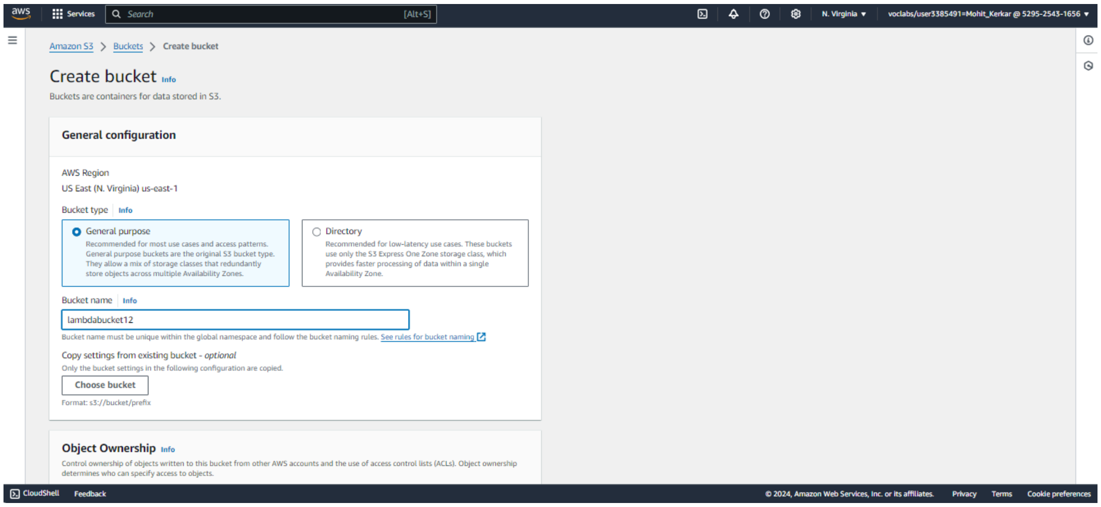
Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets (1) Info All AWS Regions

Buckets are containers for data stored in S3.

	Name	AWS Region	IAM Access Analyzer	Creation date
○	elasticbeanstalk-us-east-1-529525431656	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 5, 2024, 14:26:01 (UTC+05:30)

Give some suitable name to the bucket


aws
Services
Search [Alt+S]
N. Virginia ▾
voclabs/user3385491=Mohit_Kerkar @ 5295-2543-1656 ▾
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Amazon S3 > Buckets > Create bucket

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region

US East (N. Virginia) us-east-1

Bucket type | Info

● General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

○ Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name | Info

lambdabucket12

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

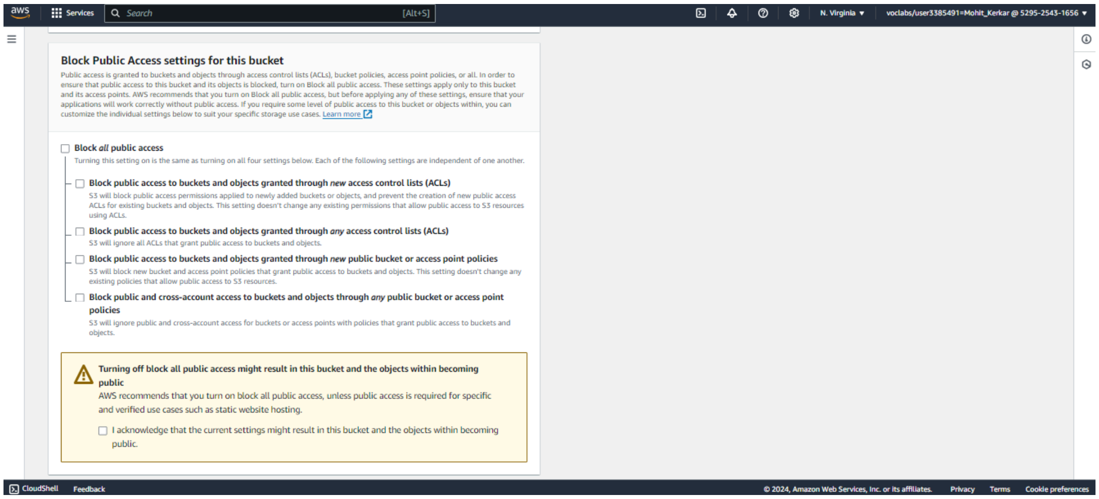
Choose bucket

Format: s3://bucket/prefix

Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

Unselect all of the default settings related to 'Block public access'


aws
Services
Search [Alt+S]
N. Virginia ▾
voclabs/user3385491=Mohit_Kerkar @ 5295-2543-1656 ▾
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

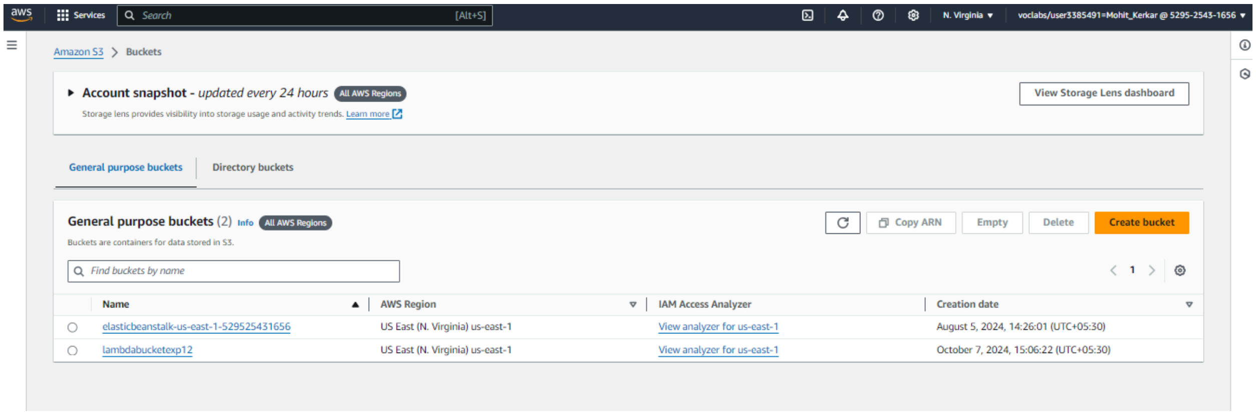
□ Block *all* public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- □ Block public access to buckets and objects granted through *new* access control lists (ACLs)
  S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- □ Block public access to buckets and objects granted through *any* access control lists (ACLs)
  S3 will ignore all ACLs that grant public access to buckets and objects.
- □ Block public access to buckets and objects granted through *new* public bucket or access point policies
  S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- □ Block public and cross-account access to buckets and objects through *any* public bucket or access point policies
  S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

⚠ Turning off block all public access might result in this bucket and the objects within becoming public
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

□ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

And here, you have your new S3 bucket created…


aws
Services
Search [Alt+S]
N. Virginia ▾
voclabs/user3385491=Mohit_Kerkar @ 5295-2543-1656 ▾
View Storage Lens dashboard
Copy ARN
Empty
Delete
Create bucket
1

Amazon S3 > Buckets

▸ Account snapshot - *updated every 24 hours* All AWS Regions

Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets | Directory buckets

General purpose buckets (2) Info All AWS Regions

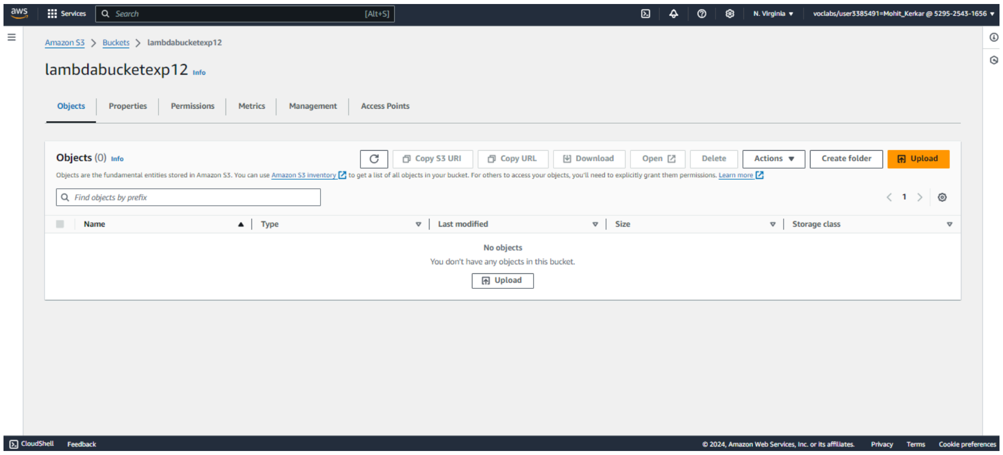
Buckets are containers for data stored in S3.

Find buckets by name

	Name ▲	AWS Region ▽	IAM Access Analyzer	Creation date ▽
○	elasticbeanstalk-us-east-1-529525431656	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 5, 2024, 14:26:01 (UTC+05:30)
○	lambdabucketexp12	US East (N. Virginia) us-east-1	View analyzer for us-east-1	October 7, 2024, 15:06:22 (UTC+05:30)

Step 2: Upload an image onto our bucket

Now, open the bucket and click on upload



lambdabucketexp12 Info

Objects (0) Info

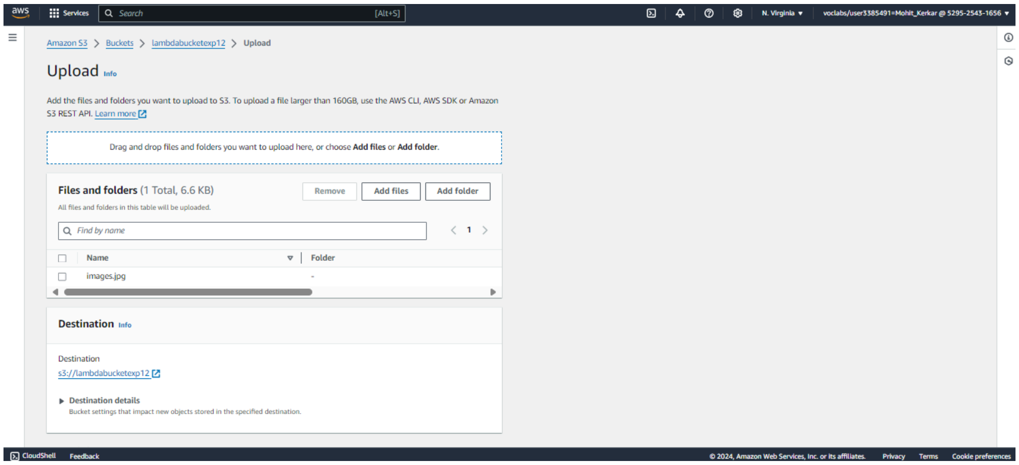
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class

No objects

You don't have any objects in this bucket.

Upload a random saved image from your device onto the bucket by clicking on 'Add files' inside the bucket


aws
Services
Search [Alt+S]
N. Virginia
voclabs/user3385491=Mohit_Kerkar @ 5295-2543-1656
Amazon S3 > Buckets > lambdabucketexp12 > Upload
Remove
Add files
Add folder
Find by name
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (1 Total, 6.6 KB)

All files and folders in this table will be uploaded.

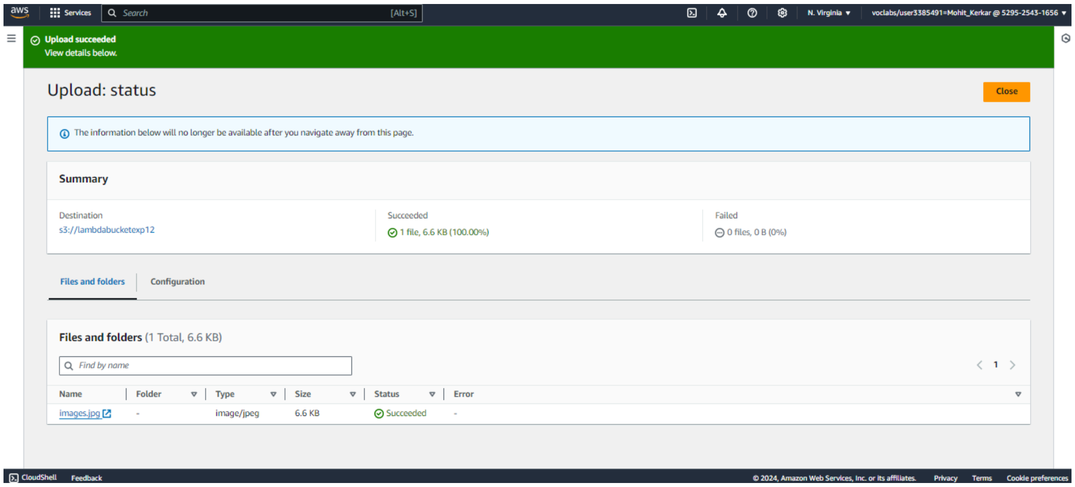
	Name	Folder
□	images.jpg	-

Destination Info

Destination
s3://lambdabucketexp12

Destination details
Bucket settings that impact new objects stored in the specified destination.

After uploading the image onto the bucket, this is the screen that will appear…



Upload succeeded
View details below.

Upload: status

The information below will no longer be available after you navigate away from this page.

Summary

Destination	Succeeded	Failed
s3://lambdabucketexp12	1 file, 6.6 KB (100.00%)	0 files, 0 B (0%)

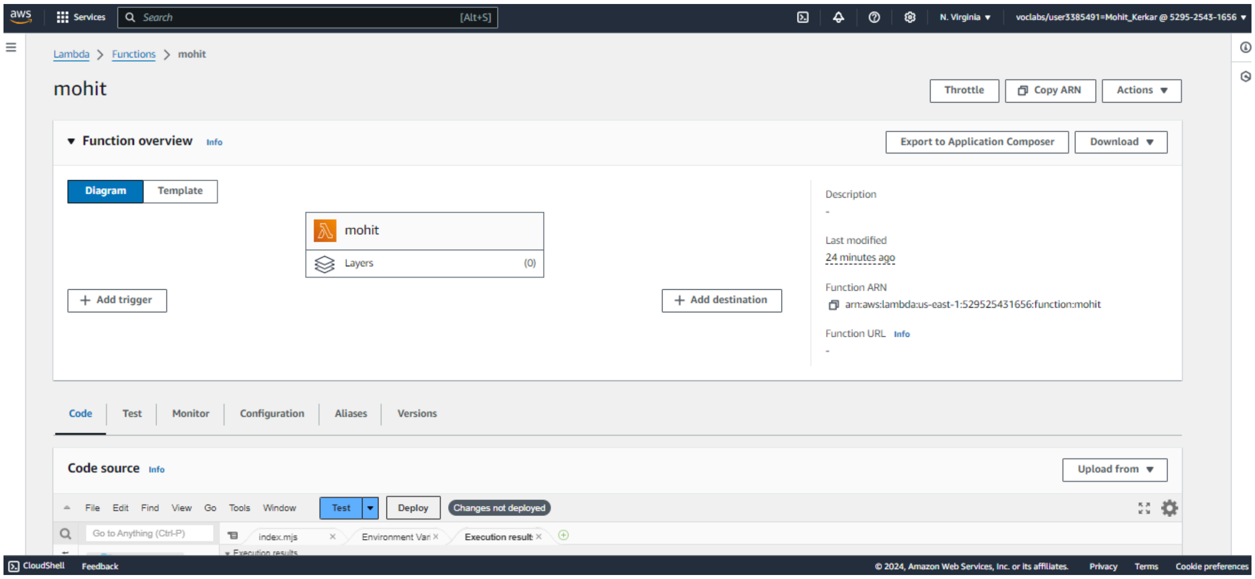
Files and folders (1 Total, 6.6 KB)

Name	Folder	Type	Size	Status	Error
images.jpg	-	image/jpeg	6.6 KB	Succeeded	-

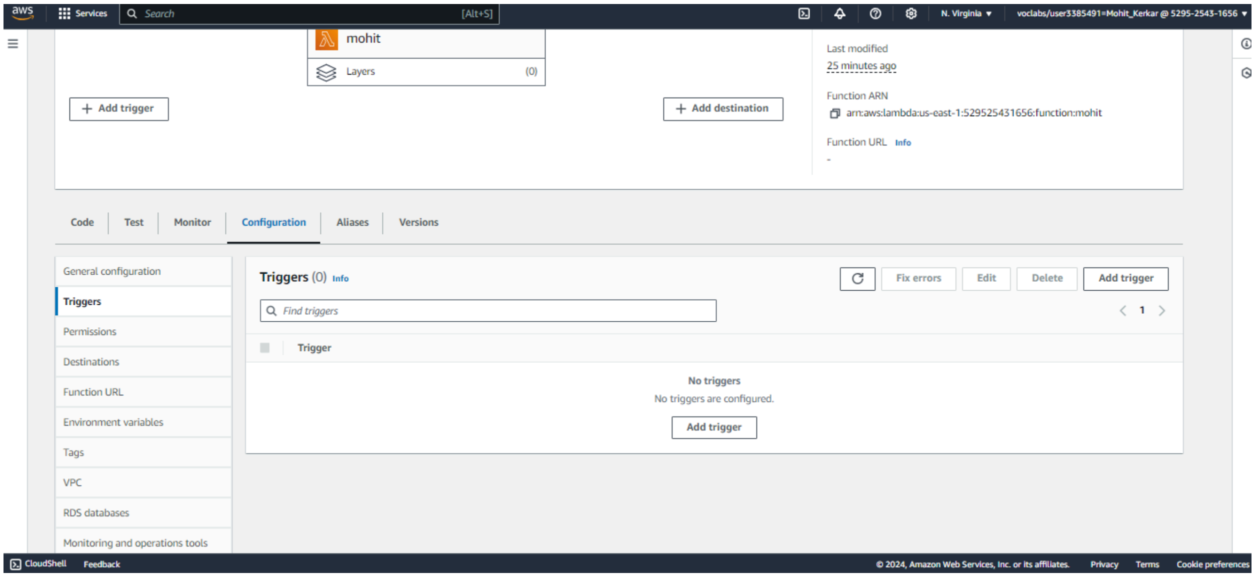
Name: Mohit Kerkar Div: D15C Roll No: 23

Step 3: Modifying Triggers inside Lambda configuration

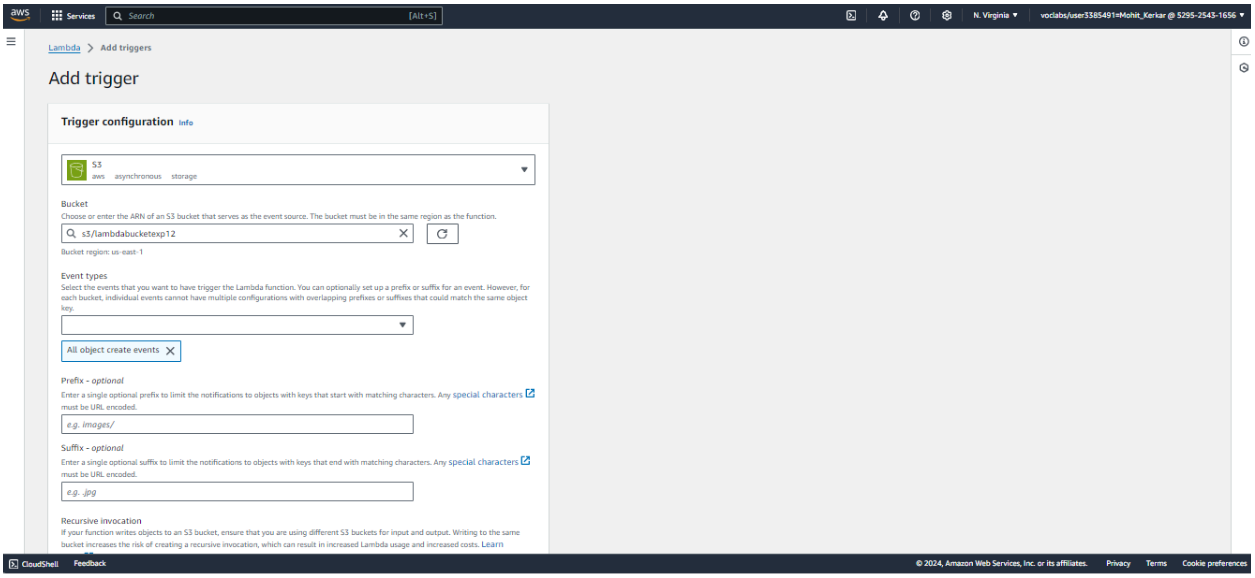
Go back to the lambda function that we created in the previous experiment



Go to the configuration section inside the function and select Triggers. This is what should appear on the screen…



Select S3 bucket inside Trigger configuration to use the S3 bucket that was created earlier



Add trigger

Trigger configuration Info

S3
aws asynchronous storage

Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.
s3/lambdabucketexp12
Bucket region: us-east-1

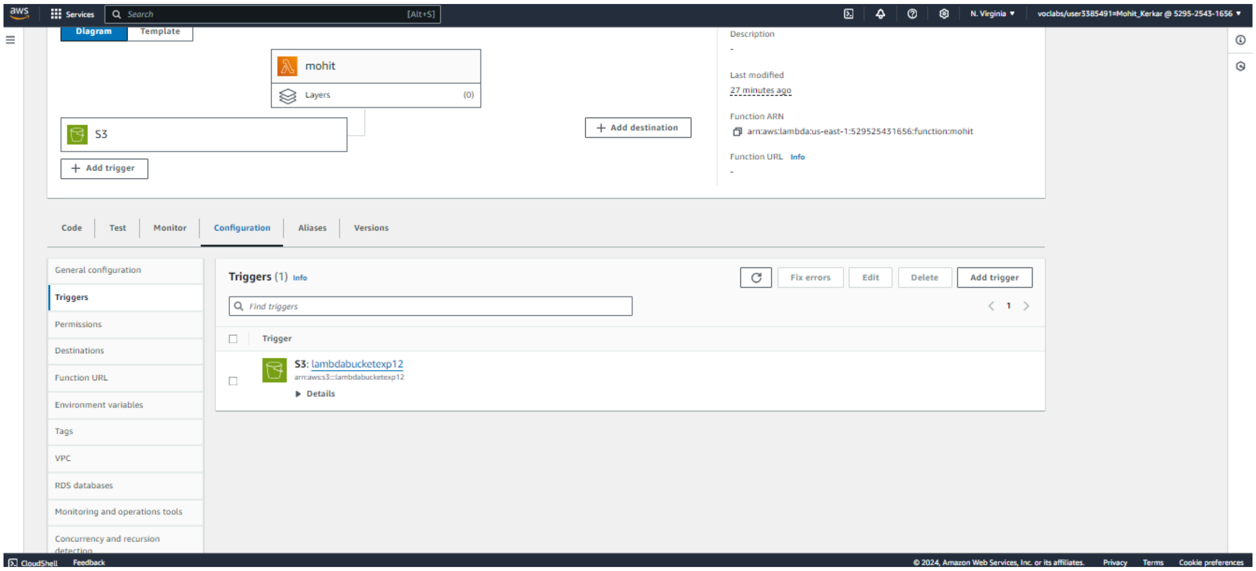
Event types
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.
All object create events

Prefix - *optional*
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any special characters must be URL encoded.
e.g. images/

Suffix - *optional*
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any special characters must be URL encoded.
e.g. .jpg

Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. Learn

After saving the changes, we see those changes being applied in the Configuration section



Description
-

Last modified
27 minutes ago

Function ARN
arn:aws:lambda:us-east-1:529525431656:function:mohit

Function URL Info
-

Triggers (1) Info

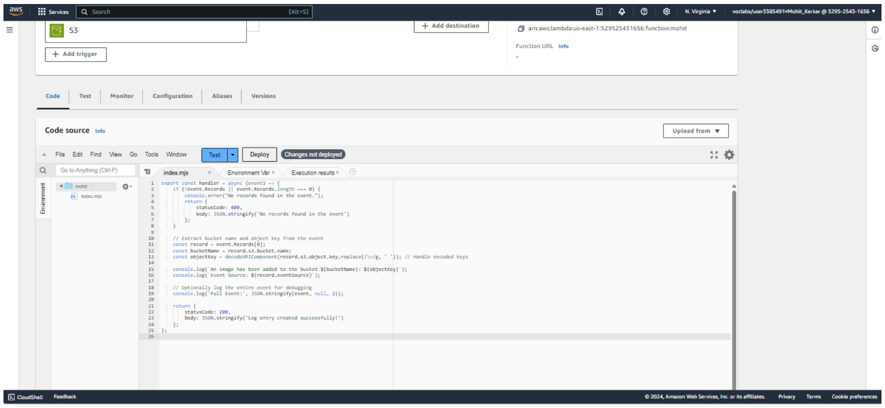
□	Trigger
□	S3: lambdabucketexp12 arn:aws:s3:::lambdabucketexp12 ▶ Details

Step 4: Testing our code

In the code section, we are replace the default code with the following code:

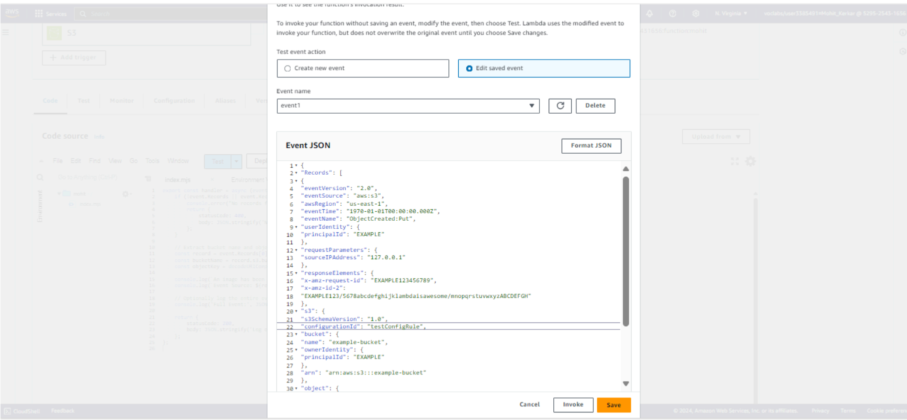
```
export const handler = async (event) => {
  if (!event.Records || event.Records.length === 0) {
    console.error("No records found in the event.");
    return {
      statusCode: 400,
      body: JSON.stringify('No records found in the event')
    };
  }
```

```
    // Extract bucket name and object key from the event
    const record = event.Records[0];
    const bucketName = record.s3.bucket.name;
    const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle
encoded keys

    console.log(`An image has been added to the bucket ${bucketName}: ${objectKey}`);
    console.log(`Event Source: ${record.eventSource}`);

    // Optionally log the entire event for debugging
    console.log('Full Event:', JSON.stringify(event, null, 2));

    return {
        statusCode: 200,
        body: JSON.stringify('Log entry created successfully!')
    };
};
```



```
export const handler = async (event) => {
    if (!event.Records || event.Records.length === 0) {
        console.error("No records found in the event.");
        return {
            statusCode: 400,
            body: JSON.stringify('No records found in the event')
        };
    }

    // Extract bucket name and object key from the event
    const record = event.Records[0];
    const bucketName = record.s3.bucket.name;
    const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle encoded keys

    console.log(`An image has been added to the bucket ${bucketName}: ${objectKey}`);
    console.log(`Event Source: ${record.eventSource}`);

    // Optionally log the entire event for debugging
    console.log('Full Event:', JSON.stringify(event, null, 2));

    return {
        statusCode: 200,
        body: JSON.stringify('Log entry created successfully!')
    };
};
```

For this particular code, we are supposed to edit the configurations for our test and save the event's JSON as we see in the following screenshot



To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.

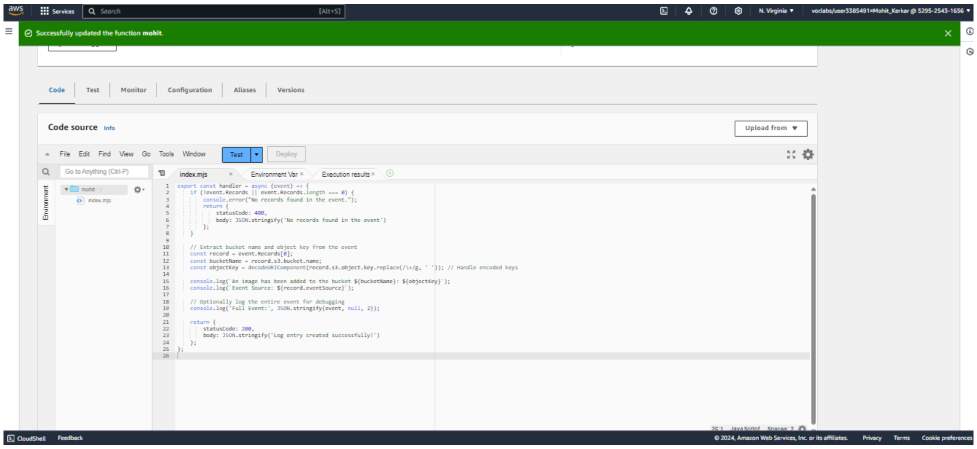
Test event action

○ Create new event ● Edit saved event

Event name: event1

```
{
"Records": [
{
"eventVersion": "2.0",
"eventSource": "aws:s3",
"awsRegion": "us-east-1",
"eventTime": "1970-01-01T00:00:00.000Z",
"eventName": "ObjectCreated:Put",
"userIdentity": {
"principalId": "EXAMPLE"
},
"requestParameters": {
"sourceIPAddress": "127.0.0.1"
},
"responseElements": {
"x-amz-request-id": "EXAMPLE123456789",
"x-amz-id-2":
"EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
},
"s3": {
"s3SchemaVersion": "1.0",
"configurationId": "testConfigRule",
"bucket": {
"name": "example-bucket",
"ownerIdentity": {
"principalId": "EXAMPLE"
},
"arn": "arn:aws:s3:::example-bucket"
},
"object": {
```

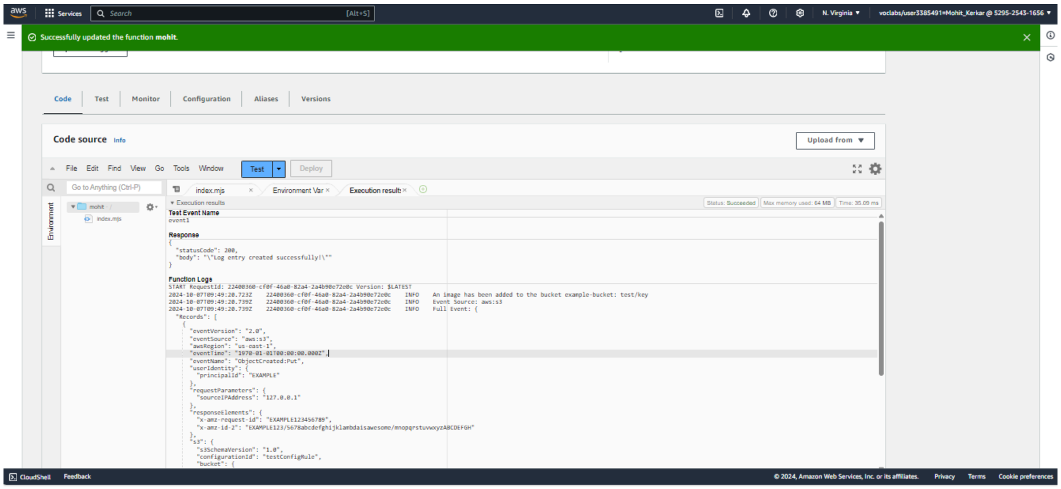
Click on deploy



Successfully updated the function **mohit**.

```
export const handler = async (event) => {
    if (!event.Records || event.Records.length === 0) {
        console.error("No records found in the event.");
        return {
            statusCode: 400,
            body: JSON.stringify('No records found in the event')
        };
    }

    // Extract bucket name and object key from the event
    const record = event.Records[0];
    const bucketName = record.s3.bucket.name;
    const objectKey = decodeURIComponent(record.s3.object.key.replace(/\+/g, ' ')); // Handle encoded keys

    console.log(`An image has been added to the bucket ${bucketName}: ${objectKey}`);
    console.log(`Event Source: ${record.eventSource}`);

    // Optionally log the entire event for debugging
    console.log('Full Event:', JSON.stringify(event, null, 2));

    return {
        statusCode: 200,
        body: JSON.stringify('Log entry created successfully!')
    };
};
```

Click on test after deploying. This is the execution result that we see. Here, we see a line being mentioned inside the execution result…. ‘An image has been added to the bucket’.

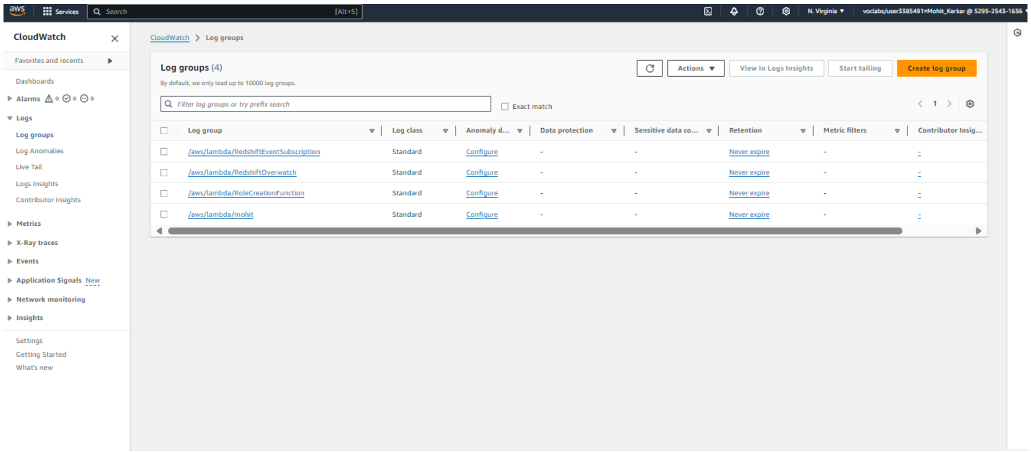


Successfully updated the function **mohit**.

```
Test Event Name
event1

Response
{
  "statusCode": 200,
  "body": "\"Log entry created successfully!\""
}

Function Logs
START RequestId: 22400360-cf0f-46a0-82a4-2a4b90e72e0c Version: $LATEST
2024-10-07T09:49:20.723Z    22400360-cf0f-46a0-82a4-2a4b90e72e0c    INFO    An image has been added to the bucket example-bucket: test/key
2024-10-07T09:49:20.739Z    22400360-cf0f-46a0-82a4-2a4b90e72e0c    INFO    Event Source: aws:s3
2024-10-07T09:49:20.739Z    22400360-cf0f-46a0-82a4-2a4b90e72e0c    INFO    Full Event: {
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
```

Step 5: Check logs

Search for CloudWatch. Click on logs -> logs groups

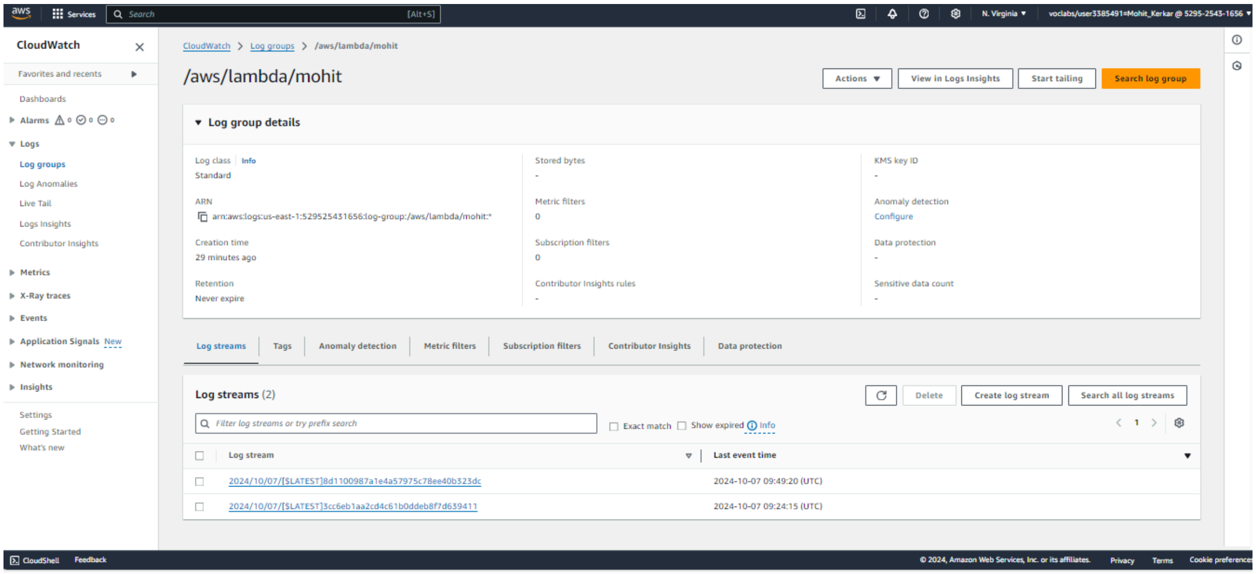


Log groups (4)

By default, we only load up to 10000 log groups.

Log group	Log class	Anomaly d...	Data protection	Sensitive data co...	Retention	Metric filters	Contributor Insig...
/aws/lambda/RedshiftEventSubscription	Standard	Configure	-	-	Never expire	-	-
/aws/lambda/RedshiftOverwatch	Standard	Configure	-	-	Never expire	-	-
/aws/lambda/RoleCreationFunction	Standard	Configure	-	-	Never expire	-	-
/aws/lambda/mohit	Standard	Configure	-	-	Never expire	-	-

In here, select the logs that we want to see



/aws/lambda/mohit

▾ Log group details

Log class | Info
Standard

ARN
arn:aws:logs:us-east-1:529525431656:log-group:/aws/lambda/mohit:*

Creation time
29 minutes ago

Retention
Never expire

Stored bytes
-

Metric filters
0

Subscription filters
0

Contributor Insights rules
-

KMS key ID
-

Anomaly detection
Configure

Data protection
-

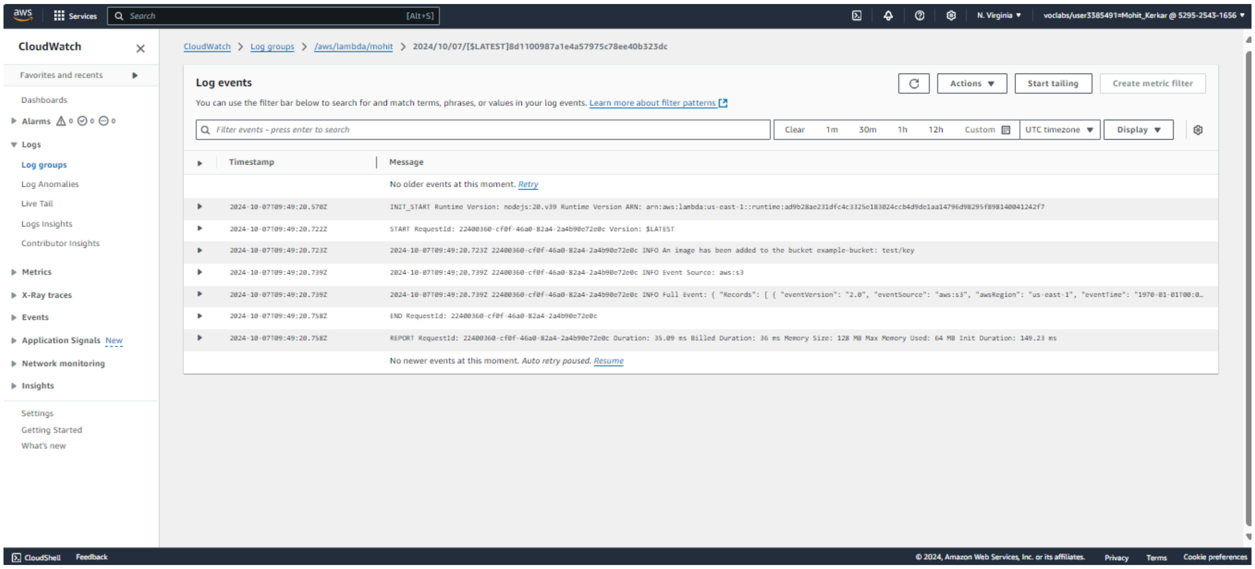
Sensitive data count
-

Log streams (2)

□ Exact match □ Show expired ⓘ Info

□	Log stream	Last event time
□	2024/10/07/[$LATEST]8d1100987a1e4a57975c78ee40b323dc	2024-10-07 09:49:20 (UTC)
□	2024/10/07/[$LATEST]3cc6eb1aa2cd4c61b0ddeb8f7d639411	2024-10-07 09:24:15 (UTC)

Here, we see it saying ‘An image has been added to bucket’, which matches the execution results in the test that we performed on our code



Log events

You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                   Message
                            No older events at this moment. Retry
2024-10-07T09:49:20.570Z    INIT_START Runtime Version: nodejs:20.v39 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:ad9b28ae231dfc4c3325e183024ccb4d9de1aa14796d98295f898140041242f7
2024-10-07T09:49:20.722Z    START RequestId: 22400360-cf0f-46a0-82a4-2a4b90e72e0c Version: $LATEST
2024-10-07T09:49:20.723Z    2024-10-07T09:49:20.723Z 22400360-cf0f-46a0-82a4-2a4b90e72e0c INFO An image has been added to the bucket example-bucket: test/key
2024-10-07T09:49:20.739Z    2024-10-07T09:49:20.739Z 22400360-cf0f-46a0-82a4-2a4b90e72e0c INFO Event Source: aws:s3
2024-10-07T09:49:20.739Z    2024-10-07T09:49:20.739Z 22400360-cf0f-46a0-82a4-2a4b90e72e0c INFO Full Event: { "Records": [ { "eventVersion": "2.0", "eventSource": "aws:s3", "awsRegion": "us-east-1", "eventTime": "1970-01-01T00:0…
2024-10-07T09:49:20.758Z    END RequestId: 22400360-cf0f-46a0-82a4-2a4b90e72e0c
2024-10-07T09:49:20.758Z    REPORT RequestId: 22400360-cf0f-46a0-82a4-2a4b90e72e0c Duration: 35.09 ms Billed Duration: 36 ms Memory Size: 128 MB Max Memory Used: 64 MB Init Duration: 149.23 ms
                            No newer events at this moment. Auto retry paused. Resume
```

Conclusion: Integrating AWS Lambda with S3 allows for real-time, automated processing of events such as file uploads. In this example, a Lambda function is configured to log a message whenever an image is added to a specific S3 bucket. This setup demonstrates the power and flexibility of serverless computing by automating tasks without requiring manual intervention or server management. By leveraging AWS Lambda, developers can efficiently handle event-driven workflows, reduce operational overhead, and quickly deploy scalable solutions that respond to specific actions within cloud environments.